矿工详细设计

目录

[**1    引言** 2](#_Toc520474906)

[**2  设计概述** 3](#_Toc520474907)

[**3  模块详细需求分析** 4](#_Toc520474908)

[**4   模块详细设计** 12](#_Toc520474909)

[**5、单元测试设计** 15](#_Toc520474910)

**1    引言**

**1.1     编写目的**

说明矿工模块与其它模块的交互和设计考虑，为程序员编码提供依据。

**1.2    背景**

开发版本：0730

开发小组：算法组

模块描述：矿工模块负责pow计算。对从验证模块接收到的交易进行Hash计算，并将结果广播到网络，由验证模块判定使用哪个矿工的结果并出块。

**1.3    参考资料**

以太坊源码

**1.4   术语定义及说明**

矿工：指参选矿工节点并且被选中的节点。

验证者：指参选验证节点并且被选中的节点。

CA：身份验证。

**2  设计概述**

**2.1   任务和目标**

1 详细说明矿工模块与其它模块之间的接口。

2 详细说明矿工模块的内部设计。

**2.1.1      需求概述**

矿工模块的主要功能是调用共识算法提供“挖矿”结果。

接收并处理ＣＡ或者自身发出的各种消息，主要包括挖矿请求，身份切换，区块到达等。

**2.1.2      运行环境概述**

Windows，Mac OS X， Linux等主流操作系统。

**2.1.3      条件与限制**

**2.1.4      详细设计方法和工具**

类图，流程图。

**3  模块详细需求分析**

**3.1     详细需求分析**

* **详细功能需求分析**

**矿工服务创建**

//HashCalcer hash calcer

type HashCalcer struct {

    engine consensus.Engine //引擎

    eth Backend

    newCACh chan testca.NewCAEvent //CA

    newCASub event.Subscription

    startMiningCh chan testca.StartMiningEvent //CA

    startMiningSub event.Subscription

    FoundCH chan event.FoundEvent

    mineSuccesSub event.Subscription

    chainHeadCh chan core.ChainHeadEvent

    chainHeadSub event.Subscription

    CurrentHeader \*types.Header

    currentCA int //-1: not found. 0: super miner 1:non-super miner 2:super verifier 3:non-super verifier

    ca \*testca.CFTEST

    Startmining func(\*types.Header, []\*big.Int)

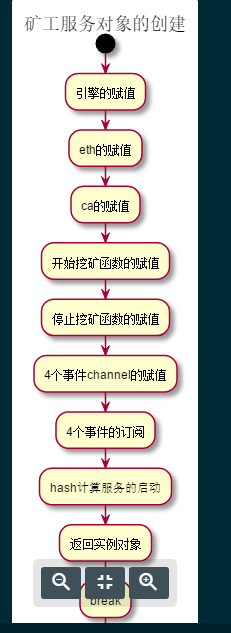
    Stopmining func()

在创建对象时对以上进行赋值，需要注意的时，4个订阅方式也要在创建对象时进行订阅。如hashcalcer.newCASub = ca.SubscribeNewCAEvent(hashcalcer.newCACh)

最后启动计算服务：go hashcalcer.update()

并返回该对象

流程图如下



**矿工模块接收到的消息：**

* 身份消息：

身份消息由CA模块发出，用于识别当前身份。

该消息包含一个整形

-1: 无身份. 0: 矿工 1:　算力输出节点 2:　验证者 3:任务分配者

矿工 -> 如果当前不是矿工则订阅挖矿请求，found， 新区块到达消息

非矿工 -> 如果当前是矿工取消之前订阅的三种挖矿消息。

* 挖矿请求消息：

由CA模块发出，通知矿工开始挖矿。矿工会订阅该消息。

该消息包括一个　types.Header

type Header struct {

    ParentHash common.Hash `json:"parentHash" gencodec:"required"`

    UncleHash common.Hash `json:"sha3Uncles" gencodec:"required"`

    Coinbase common.Address `json:"miner" gencodec:"required"`

    Root common.Hash `json:"stateRoot" gencodec:"required"`

    TxHash common.Hash `json:"transactionsRoot" gencodec:"required"`

    ReceiptHash common.Hash `json:"receiptsRoot" gencodec:"required"`

    Bloom Bloom `json:"logsBloom" gencodec:"required"`

    Difficulty \*big.Int `json:"difficulty" gencodec:"required"`

    Number \*big.Int `json:"number" gencodec:"required"`

    GasLimit \*big.Int `json:"gasLimit" gencodec:"required"`

    GasUsed \*big.Int `json:"gasUsed" gencodec:"required"`

    Time \*big.Int `json:"timestamp" gencodec:"required"`

    Extra []byte `json:"extraData" gencodec:"required"`

    MixDigest common.Hash `json:"mixHash" gencodec:"required"`

    Nonce BlockNonce `json:"nonce" gencodec:"required"`

}

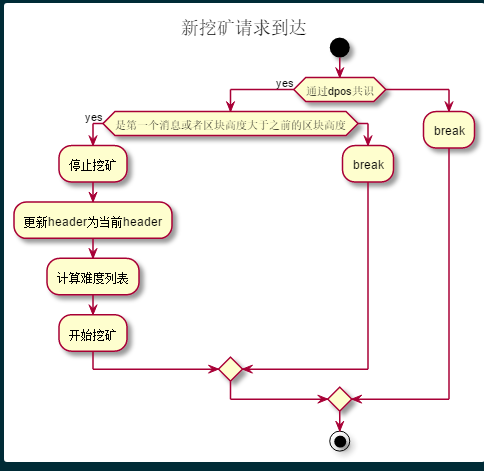
如果接收到的Ｈｅａｄｅｒ大于当前Ｈｅａｄｅｒ，并且共识成功，则开始新一轮挖矿。

具体流程为，先使用CA的dpos函数判断是否通过，不通过直接返回不做任何处理，

如果通过，判断当前是不是第一个消息到来或者当前区块的高度是否大于上一个区块的高度。如果不是，则不做任何处理。

如果通过，首先停止挖矿，更新收到的header为当前header，并计算难度列表，开始挖矿

流程图如下：



* Found消息：

由共识模块发出，当矿工调用共识模块进行ＰＯＷ计算时，无论结果如何，共识模块会将该消息发出。该消息包含结果的难度值，类型为\*big.Int　和\*types.Header。

如果难度值与当前保存Header的难度值一致（小于？？）则停止挖矿。

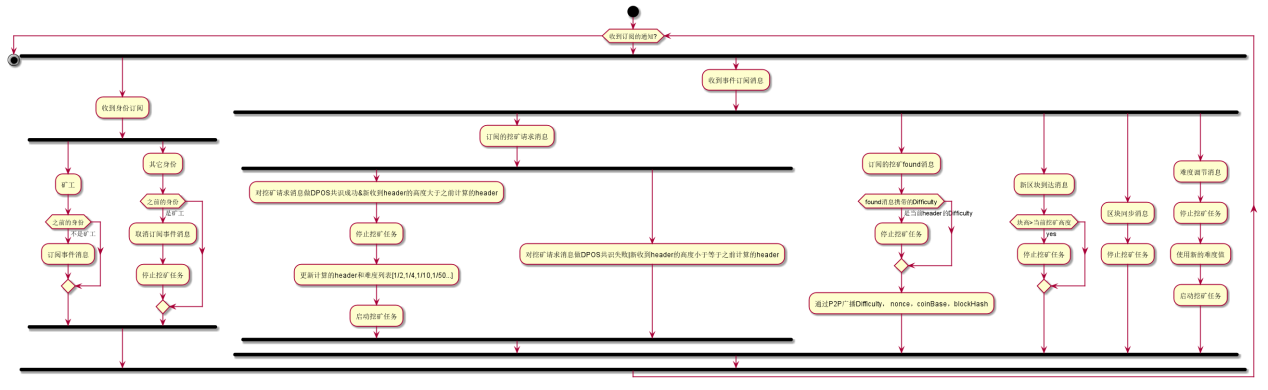
* 新区块到达消息：

该消息由区块同步模块发出，当下载一个新区块成功后，发出该消息。

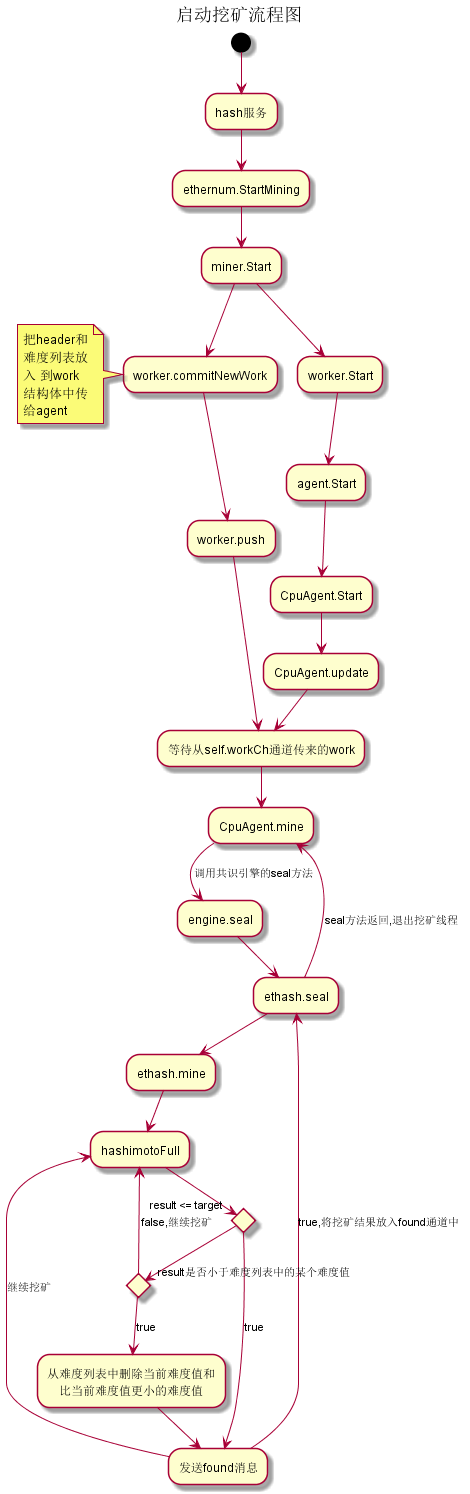
新区块高度一定大于等于当前挖矿高度，应停止挖矿。

* 同步消息：

节点刚上线时检查当前数据库与网络区块高度差，如果需要同步，则不能挖矿，因为本地当前节点高度太低。



**加入难度列表的挖矿流程:**



### 交易打包处理和blockchain插入

以太坊中这部分代码在worker.wait 函数中处理。在新的流程中把wait函数中的代码已经注释掉。挖出矿后再 CpuAgent.mine函数中退出，不会走到wait函数中。因此也不会有交易打包操作和blockchain插入操作。

### header生成 去掉

以太坊中在ethash.mine 函数中会创建header，在新版本中把这部分代码注释掉。使用从hash服务传进来的header。

* **详细性能需求分析**

矿工模块应具备多线程挖矿能力。多个线程同时进行ＰＯＷ计算， 用户可以添加矿工（Agent）。（接到挖矿消息时，是否需要同时起多个矿工）

* 详细系统运行环境及限制条件分析

POW算法通过进行CPU挖矿，且内存需要大于2G。

**3.2         详细系统运行环境及限制条件分析接口需求分析**

* 系统接口需求分析

系统初始化后，由主程序调用矿工开始进行消息监听并处理监听到的消息。对外部开放接口包括：

矿工初始化接口： 传入矿工所需的对象，比如共识引擎对象，配置管理对象，消息发布对象等

矿工监听消息接口：矿工初始化完成后，可由外部启动或关闭监听。

矿工状态接口： 返回当前矿工的工作状态。正在挖矿还是没在挖矿。

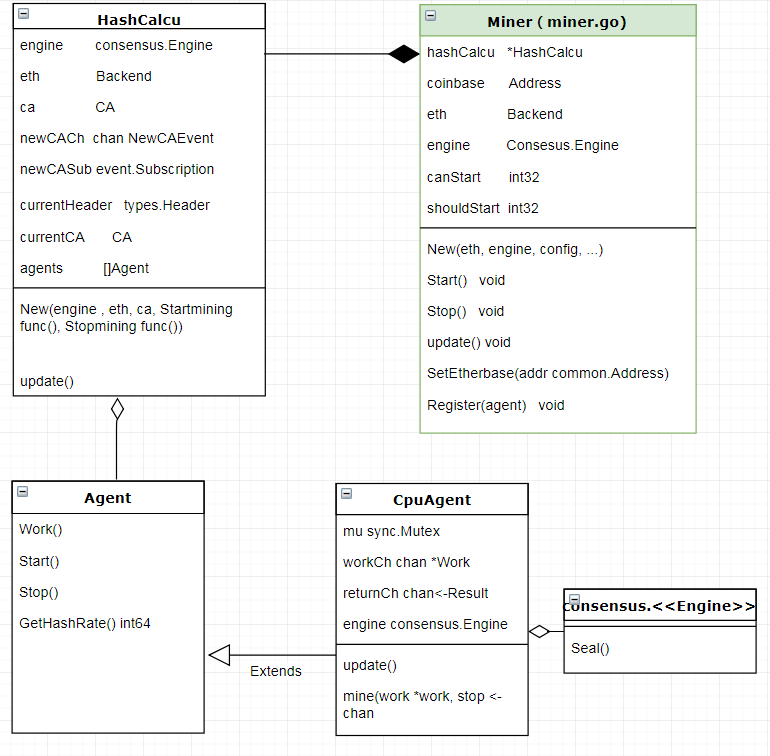
设置CoinBase接口： 设置矿工的CoinBase。

添加Agent接口 （NewPublicMinerAPI）： 增加一个Agent并行挖矿。

由于需要接收各类消息，因此要求CA模块，共识模块，区块下载模块提供消息订阅接口供矿工模块订阅。消息订阅模式采用以太坊Event包中的Feed类。

**4   模块详细设计**

**4.1    模块结构设计**



**4.2    功能详细设计**

**功能名称：update（）**

**输入：监听的各种消息**

**处理：**

func (self \*HashCalcer) update() {

    defer self.newCASub.Unsubscribe()

    defer self.startMiningSub.Unsubscribe()

    defer self.mineSuccesSub.Unsubscribe()

    defer self.chainHeadSub.Unsubscribe()

    for {

        select {

        case data := <-self.chainHeadCh:

            fmt.Println("Message Comming：新区快到达")

            self.chainHeadHandle(data)

        case data := <-self.newCACh:

            fmt.Println("Message Comming：新身份到达")

            self.newCAHandle(data)

        case data := <-self.startMiningCh:

            fmt.Println("Message Comming:新挖矿请求到达")

            var h \*types.Header

            testHeader := types.Header{

                ParentHash: common.HexToHash("0xd783efa4d392943503f28438ad5830b2d5964696ffc285f338585e9fe0a37a05"),

                Difficulty: big.NewInt(600000),

                Number: big.NewInt(3311058),

            }

            h = &testHeader

            data.Header = h

            self.startMiningHandle(data)

        case data := <-self.FoundCH:

            fmt.Println("Message Comming：Found事件到达", data)

            self.FoundHandle(data)

        }

    }

}

**算法描述：**

* 身份消息：

身份消息由CA模块发出，用于识别当前身份。

该消息包含一个整形

-1: 无身份. 0: 矿工 1:　算力输出节点 2:　验证者 3:任务分配者

矿工 -> 如果当前不是矿工则订阅挖矿请求，found， 新区块到达消息

非矿工 -> 如果当前是矿工取消之前订阅的三种挖矿消息。

* 挖矿请求消息：

由CA模块发出，通知矿工开始挖矿。矿工会订阅该消息。

该消息包括一个　types.Header

如果接收到的Ｈｅａｄｅｒ大于当前Ｈｅａｄｅｒ，并且共识成功，则开始新一轮挖矿。

* Found消息：

由共识模块发出，当矿工调用共识模块进行ＰＯＷ计算时，无论结果如何，共识模块会将该消息发出。该消息包含结果的难度值，类型为\*big.Int　和\*types.Header。

如果难度值与当前保存Header的难度值一致（小于？？）则停止挖矿。

* 新区块到达消息：

该消息由区块同步模块发出，当下载一个新区块成功后，发出该消息。

新区块高度一定大于等于当前挖矿高度，应停止挖矿。

**输出：无**

**5、单元测试设计**

1. 矿工微服务创建

|  |  |
| --- | --- |
| 测试项目 | Minner-Micro service 矿工微服务创建 |
| 用例编号 | Minner-Micro service-TC001 |
| 用例名称 | 矿工为服务的创建 |
| 用例目的 | 验证矿工服务对象是否能正常创建，hash计算服务是否开启，订阅的事件能否收到 |
| 测试环境 | 一台pc机器 |
| 预置条件 | 无 |
| 测试过程 | 1. 创建eth对象，eth对象会创建miner对象，miner对象里会创建矿工为服务对象，故，创建eth对象 2. 定时模拟4中消息的发送，观察hash计算中能否收到订阅的消息 |
| 预期结果 | 1. 对象创建成功，也成功掉起update(hash计算服务开启) 2. 模拟的4种方式均能收到（因为这4种消息开源于3个地方CA,blcokchain和sealer。所以模拟消息发送写的也比较乱，但是能订阅成功，也能正常接收数据） |

1. 新挖矿请求到来的处理

|  |  |
| --- | --- |
| 测试项目 | New-StartMining-Handle 新挖矿请求到来的处理 |
| 用例编号 | New-StartMining-Handle-TC0001 |
| 用例名称 | 新挖矿请求到来的处理 |
| 用例目的 | 当收到新挖矿请求时能否正常处理 |
| 测试环境 | 一台pc机器 |
| 预置条件 | 无 |
| 测试过程 | 1发送一个包含header的挖矿请求  2 再发送一个比上一个header高度小的header  3 再发送一个比第一个header高度大的header |
| 预期结果 | 1 当收到第一个header到来时(因为时第一个所以特殊处理)，停止挖矿，更新为当前header，更新难度列表来世挖矿  2 当收到第二个header到来时，发现header高度小于等于当前header不做处理  3 当收到第三个header到来时，发现header高度大于 当前header，停止挖矿，更新为当前header，更新难度列表，开始挖矿 |

1. 启动停止挖矿功能

|  |  |
| --- | --- |
| 测试项目 | Minner-Mining启动停止挖矿功能 |
| 用例编号 | Minner-Mining-TC001 |
| 用例名称 | 挖矿功能的启动停止 |
| 用例目的 | 验证挖矿功能能否正常启动停止 |
| 测试环境 | 一台pc机器 |
| 预置条件 | 无 |
| 测试过程 | 1. 调用启动挖矿的函数，观察是否有挖矿的打印信息 2. 调用停止挖矿函数，观察是否有停止挖矿的调试信息。 |
| 预期结果 | 1. 挖矿功能正常启动，有启动挖矿的信息打印出来 2. 挖矿功能正常停止，有停止挖矿的信息打印出来 |

1. 加入难度列表的挖矿

|  |  |
| --- | --- |
| 测试项目 | Minner-Mining 加入难度列表的挖矿 |
| 用例编号 | Minner-Mining-TC002 |
| 用例名称 | 加入难度列表的挖矿 |
| 用例目的 | 在加入难度列表后能否正常挖矿 |
| 测试环境 | 一台pc机器 |
| 预置条件 | 无 |
| 测试过程 | 1 调用启动挖矿函数时，传入header和难度列表  2 观察是否正常启动挖矿  3 观察挖出矿后打印的难度信息是否是难度列表中的难度信息。  4 观察在挖出难度列表中难度信息的矿时，是否发送found消息，是否继续挖矿。  5 观察能否直到挖到难度值为header中的难度值的矿后发送found消息退出挖矿线程。 |
| 预期结果 | 1 开始挖矿后打印header信息和难度列表，和传入的header和难度列表一致。  2 挖矿正常启动  3 首先挖出的难度值是难度列表中的难度值。  4 挖出矿后会发送found消息，并继续挖矿  5 最后挖出难度值等于header中的难度值时才，会发送found消息并退出挖矿线程。 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 类名 | | | | *HashCalu* | | | |
| 函数名 | | | | update | | | |
| 函数描述 | | | | 接收订阅的消息并处理 | | | |
| 输入/输出 | | | | 输入：无 输出：无 | | | |
| **测试用例ＩＤ** | **测试用例描述** | **预期结果** | | **实际结果** | **备注** |
| 1 | 发送非矿工身份消息 | 接收到消息并停止挖矿，注销已订阅的消息。 | |  |  |
| 2 | 发送矿工消息 | 接收到消息，注册消息。 | |  |  |
| 3 | 发送挖矿消息，Header块高大于之前高度 | 接收到消息，开始挖矿 | |  |  |
| 4 | 发送挖矿消息，Header块高小于之前高度 | 接收到消息，不处理 | |  |  |
| 5 | 发送found消息，难度值与当前匹配 | 接收到消息，停止挖矿 | |  |  |
| 6 | 发送found消息，难度值与当前不匹配 | 接收到消息，不处理 | |  |  |
| 7 | 发送新区块到达消息 | 接收到消息，停止挖矿 | |  |  |